Interview Task: LLM-Powered Chatbot with SQL Integration

Objective

Design and build a chatbot powered by a Large Language Model (LLM) (e.g., OpenAI GPT, Mistral, etc.) that:
1. Takes user input as natural language.
2. Identifies the intent and appropriate SQL table.
3. Either:
   - Generates SQL queries from the input using the LLM, OR
   - Uses Retrieval-Augmented Generation (RAG): fetches relevant rows from SQL and sends as context to the LLM.
4. Returns a conversational response to the user.

Example Use Cases

User Input	Expected Behavior
What is the status of my order ORD5678?	LLM recognizes it's about orders → queries order_status table
Give me the price of Galaxy S23.	LLM matches to product_info → returns price
I need tech support contact details.	LLM maps to support_contacts table
How do I reset my password?	LLM uses FAQ data or fallback logic

Requirements

1. SQL Database

Design and use the following tables with sample data:
- product_info(product_id, name, features, price)
- order_status(order_id, customer_name, status)
- support_contacts(department, phone, email)
- faq(id, question, keywords, answer)

2. LLM Usage (Compulsory)

Choose one or more of the following:

- 🔹 Option A: SQL Generation using LLM
- 🔹 Option B: RAG-style Retrieval

3. App Behavior

Take user question via CLI or simple UI. Call the LLM to:
- Interpret question
- Generate SQL or guide retrieval
Run SQL and return a formatted response.

🧰 Tech Stack Suggestions

Component	Tools
LLM	OpenAI GPT-3.5 / Mistral / LLaMA2
Backend	Python (Flask / FastAPI)
SQL	SQLite / MySQL / PostgreSQL
Prompting	langchain (optional)
UI	Streamlit / CLI

✨ Bonus Tasks

- Implement retry logic for SQL generation failures
- Add natural rephrasing of raw DB output
- Support follow-up questions using session memory
- Deploy the app (on Streamlit, Render, or HuggingFace Spaces)

🚀 Deliverables

- 🧠 Working code (GitHub repo or ZIP)
- 🗂 Sample SQL dump or DB creation script
- 🧾 README.md with clear instructions
- 📝 Describe how LLM is used (prompt example, retry logic, fallback plan)

✅ Evaluation Criteria

Criteria	Weight
LLM Integration & Prompting	40%
SQL Table Selection & Query Accuracy	25%
Code Structure & Modularity	15%
Response Quality (Naturalness)	10%
Bonus (RAG, UI, Deployment)	10%

📬 Submission

- Deadline: 06/04/2025 (6 : 00 PM)
- Format: GitHub link or ZIP
- Must include instructions to run locally with mock data